Johnson & Wales University

Providence, Rhode Island

College of Engineering and Design

Master of Science in Information Security and Assurance

**RESEARCH ON SECURITY OF DOCKER CONTAINERS & KUBERNETS CLUSTERS**

**By : Vishal Ojha**

**Guided By Catalan, Brandon, Tiwari, Sunil & Ide, Mary**

December 13, 2018

Table of Contents

Introduction...................................................................................................... 2

Problem Statement .......................................................................................... 3

Background of Study…………………………………………………………4

Security Audit & Concerns …………………………………………………12

Conclusion...................................................................................................... 15

Research Question…......................................................................................18

Methodology...................................................................................................20

Definition of Terms………………………………………………………….22

References.......................................................................................................23

Appendix B……………………………………………………………….…25

**List of Tables and Figures**

Figure 1.0: Virtual Machine Hypervisor 7

Figure 1.1: Type 2 Hypervisor Functioning 9

Figure 1.2: Docker Container Architecture 9

Figure 1.3: Kubernetes Orchestration Architecture 9

**Introduction**

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers (OpenSource,2017). Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. The Docker ecosystem has come a long way in 2017. Especially in security, we saw new features being launched, new products entering the market, and new best practices emerge. All this point to a 2018 that’s set to build on this progress and enable running extremely secure container workloads. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. Though a relative new technology there are very few articles and research as to how can we secure the docker container that are coming in the market. With the sudden increase in DevOps technology there is a need to explore the security concerns related to Docker container and shipping of them in a proper fashion. In this research we will discover the possible issues and how to mitigate them. We will be using docker container that are shipped by Avaya India Pvt. Ltd. & Nginx Web Server for ubuntu and Linux based docker containers. Mary Ellen-Ide, Sr. Network Security Administrator at Johnson and Wales in sync will do a full-blown Nessus audit with audits by Mr. Sunil Tiwari, Sr. Engineering Manager Avaya India Pvt. Ltd.

**Problem Statement**

Information Technology is an ever-developing sector with new software rolling out each day and multiple projects being processed daily. Once these projects are completed and rolled out in the market as initial build, these projects turn out to be an open invitation to multiple vulnerabilities across the many different Hardware and Software platforms. This generally happens because of a fault in the software development that can’t handle security breach attacks due to the traditional model of software development.

Docker is a computer program that performs operating-system-level virtualization, also known as "containerization". It was first released in 2013 and is developed by Docker, Inc. Docker is used to run software packages called "containers". (What is Docker? ,2017) Since then Docker has been an industry favorite for deployment of packages and the application associated with the package. A January 2017 analysis of LinkedIn profile mentions showed Docker presence grew by 160% in 2016.The software has been downloaded more than 13 billion times as of 2017 (Wikipedia,2018). The development officials at Avaya found out the Docker though useful doesn’t have a lot of security features either listed out as a white paper or scanned and exposed.

Docker being more popular than a Hyper Visor or a VM in the industry there is a need to find out more secure measures with the Docker. A survey by “RightScale” observed that 49% people have adopted to docker based cloud infrastructure in 2018 than the 35% in 2017.(ZD Nett, 2018) With such a whooping increase in the number of users it becomes necessary to determine the security issues and the ways to make Docker Container and Kubernetes Cluster secure and user friendly.

**Background of the Study**

Multiple systems have different specification for how it works and what kind of software it needs to make it efficient to use. In traditional method that would be creating separate system with specific software set loaded to perform a specific function. The special feature of loading multiple OS in one system that came along side Windows NT and 98 made it easy to alter the Master Boot record and adding two OS in one made it a little easy and cost efficient. Comes 1998 when VMware finally became popular making its impression in public making it a public favorite to develop and run multiple OS on a single machine. At the operating system level, it can only virtualize one OS: the guest OS is the host OS. This is like having many terminal server sessions without locking down the desktop. Thus, this is the best of both worlds, having the speed of a TS session with the benefit of full access to the desktop as a virtual machine, where the user can still control the quotas for CPU, RAM and HDD. Similar to the hardware level, this is still considered a Server Virtualization where each guest OS has its own IP address, so it can be used for networking applications such as web hosting. The major issue with these virtual machines was though they make it possible to run multiple applications on the same physical hardware while keeping conflicts among software components and competition for hardware resources to a minimum they are bulky—typically gigabytes in size. They don’t really solve problems like portability, software updates, or continuous integration and continuous delivery. (InfoWorld,2018). Along with the developing data method the VM also got accompanied by Hypervisor another tool that made its way in the heart of the server teams making them a favorite for server service deployment. Below Figure 1.0 & 1.1 shows how Hypervisor architecture works.

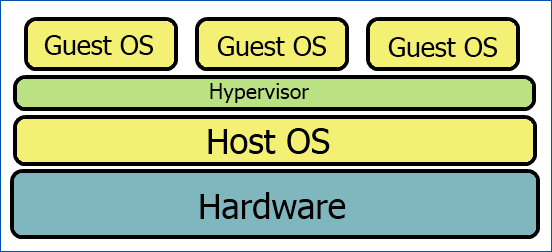
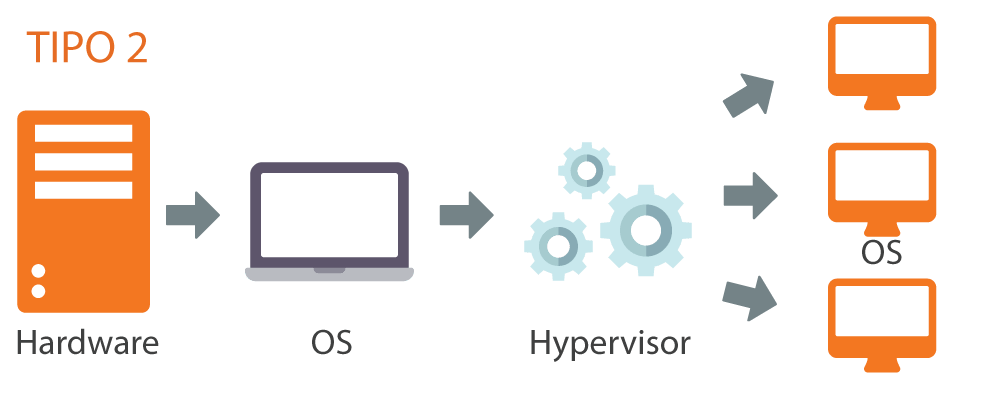
Figure 1.0: Virtual Machine Hypervisor

Figure 1.1: Type 2 Hypervisor functioning

The server needs to load the Hypervisor client which the hypervisor in turn triggers the Necessary OS client. Depending on the needs of the system the Hypervisor allocates necessary resources like memory, disk storage, bandwidth and other resources that the system may use to get around and make it work appropriately. The hypervisor is itself classified by the way it functions namely

1. Bare metal or Type 1 hypervisor B. Embedded or hosted or Type 2 Hypervisor.
2. ***Bare metal or Type 1 Hypervisor.***

In this type the Hypervisor is an independent machine, it has its own operating system which is installed on its own hardware and this in turn creates a virtualization layer on which the virtual machines with different machines like Windows, OS X or Linux are created. The VM Ware ESXI server is an example of this. In the ESXi model of Hypervisor the Linux kernel is the primary virtual machine; it is invoked by the service console. At normal run-time, the vmkernel is running on the bare computer, and the Linux-based service console runs as the first virtual machine. (Wikipedia,2017)

1. ***Embedded Hypervisor or Type 2***

The most popular hypervisor among the people is the type 2 hypervisor also known as embedded hypervisor. The hypervisor has been in use since early 2017 and is popular among individuals and small organization developing software or servers. The way it functions is the virtual machine is a software that is embedded on top of a host machine with its own operating system. So, there is a Host Machine running it’s independent Operating System on to which a VM Software is installed with its own virtual operating system and on top of that is installed different set of virtual machines.

**Docker: The way to change it all.**

Enter Docker containers. Containers make it possible to isolate applications into small, lightweight execution environments that share the operating system kernel. Typically measured in megabytes, containers use far fewer resources than virtual machines and start up almost immediately. They can be packed far more densely on the

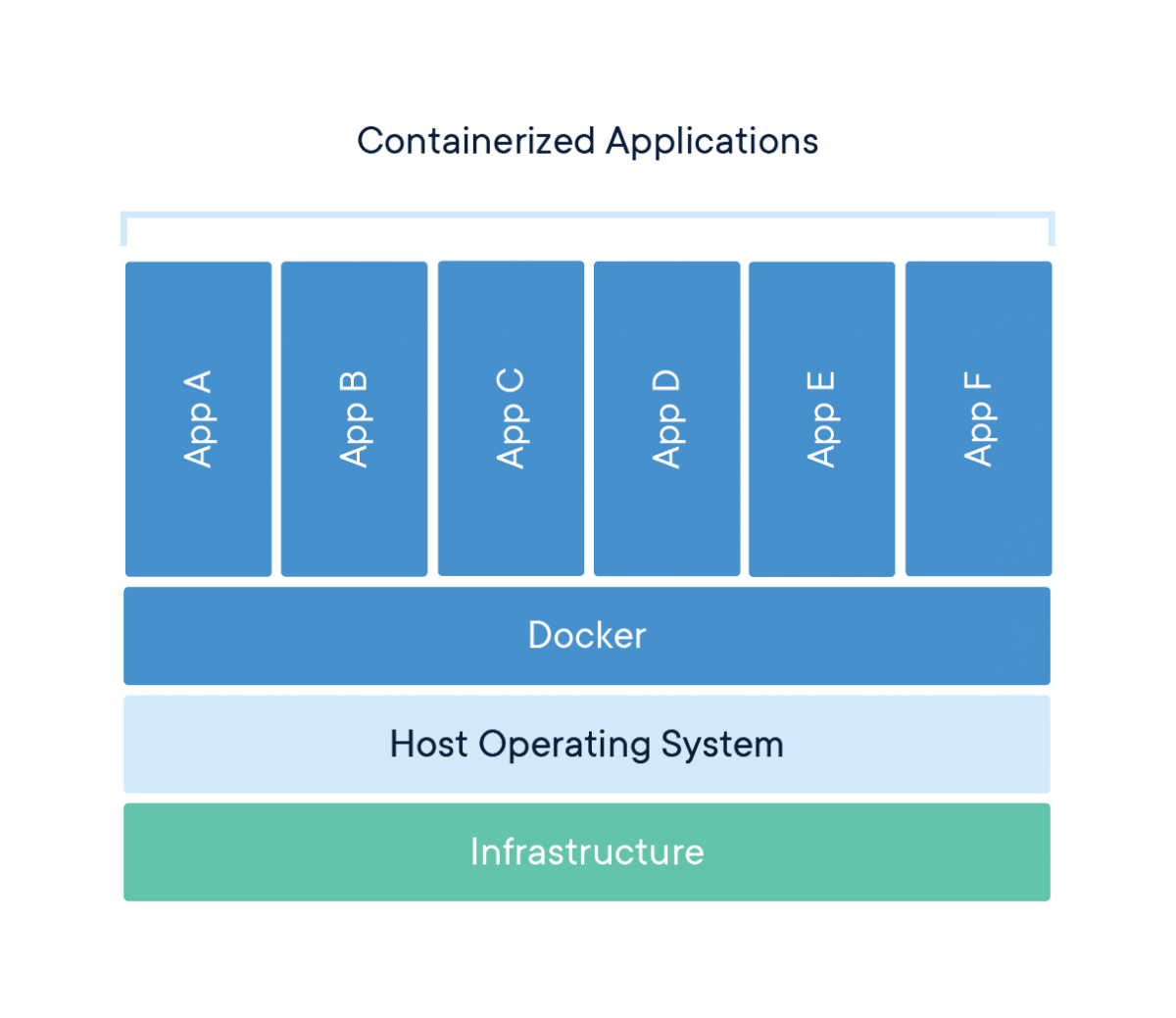
same hardware and spun up and down en masse with far less effort and overhead. (InfoWorld,2018). Figure 1.1 show how the docker container architecture works.

Fig 1.2: Docker Container Architecture

Mainly, the Docker container is designed using the Linux BDS or the Solaris to support the data image being built on top of the systems. Containers are a concept at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as secluded procedures in user space. Containers take up less space than VMs (container images are usually tens of MBs in size), can hold more requests and require fewer VMs and Operating systems. This makes the containers a very easy and helpful piece of equipment to send multiple images of the system configuration without

manually configuring all the data and the applications. PHP, Ruby, Java, and Nodejs. are the main programming frameworks used in containers.

In a recent survey by a website named DevOps Zone it was seen and quoted “2/3 of companies that try using Docker, adopt it”. (DZone.com, 2017). Most businesses who will adopt have already done so within 30 days of initial production usage, and nearly all the residual adopters adapt within 60 days. Docker adoption is up 30% in the last year. Adopters multiply their containers by five each day and docker adopters approximately quintuple the average number of running containers they have in production between their first and tenth month of usage.

**Docker: Advantage and Disadvantages**

Although a relatively new technology docker gained a lot of popularity and usage making it a recent industry favorite. Some of the advantages are below.

1. Standardization and Productivity
2. CI Efficiency
3. Rapid Deployment
4. Flexibility of the image
5. Multi cloud platform support
6. Isolation

Though these advantages surely make it one of the most important piece of available software, there are certain disadvantages that comes its way.

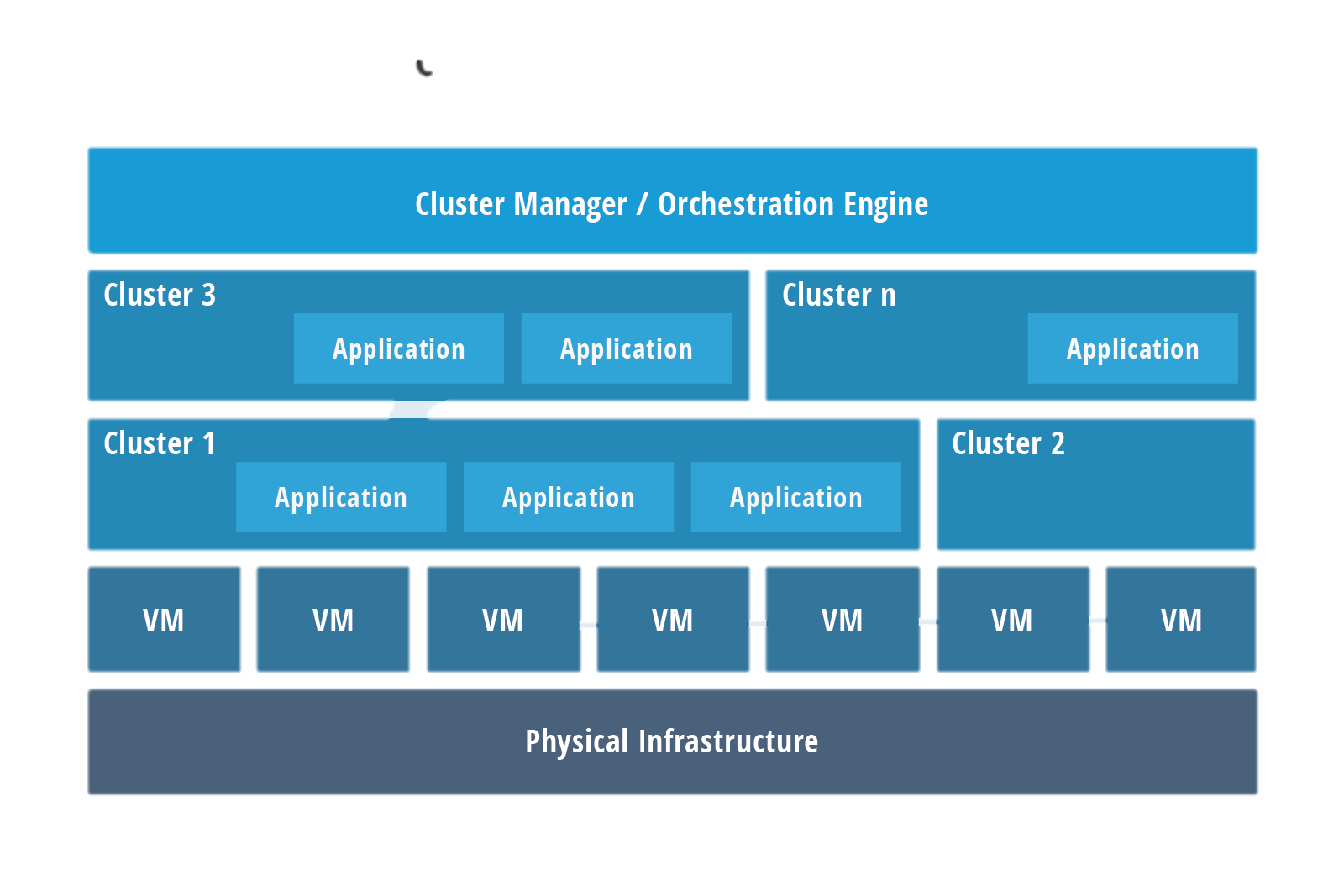
1. Containers do not keep running at bare metal rates. Holders devour assets more proficiently than virtual machines. Nevertheless, containers are yet subject to execution

overhead because of overlay networking, interfacing among docker containers and the host framework, etc. On the off chance, that you need 100 percent bare metal execution, you have to utilize bare metal, not containers.

1. The containers biological system is cracked. Despite the fact that the center Docker stage is open source, some docker container items don't work with different ones - generally because of rivalry between the organizations that back them For example, OpenShift, Red Hat's container-as-a-service platform, only works with the Kubernetes orchestrator.
2. Tireless information stockpiling is convoluted. By plan, the majority of the information inside a container vanishes always when the container close down, except if you spare it elsewhere first. There are approaches to spare information diligently in Docker, such as Docker Data Volumes, but this is arguably a challenge that still has yet to be addressed in a seamless way.
3. Graphical applications don't function admirably. Docker was structured as an answer for conveying server applications that don't require a graphical interface. While there are some inventive systems, (for example, X11 video sending) that you can use to run a GUI application inside a container, these arrangements are awkward, best case scenario.
4. Not all applications profit by containment. When all is said in done, just applications that are intended to keep running as an arrangement of attentive micro services remain to pick up the most from containers. Something else, Docker's solitary genuine advantage is that it can streamline application conveyance by giving a simple bundling system.

**Kubernetes Cluster a.k.a Kubernetes Orchestration**

Orchestration is the automated arrangement, coordination, and management of computer systems, middleware, and services. While Docker gave an open standard to bundling and

dispersing containerized applications, there emerged another issue. How might these containers be composed and planned? How do all the diverse holders in your application speak with one another? By what means can container occurrences be scaled? Answers for organizing containers before long developed. Kubernetes, Mesos, and Docker Swarm are a portion of the more prevalent alternatives for giving a deliberation to influence a group of machines to act like one major machine, which is essential in a huge scale condition. The below Fig 1.3 shows the Kubernetes cluster and its architecture as compared to that of Docker.

Container Orchestration Architecture.

**Docker vs Kubernetes Cluster**

Kubernetes and Docker are both far reaching true answers for cleverly oversee containerized applications and give incredible abilities, and from this some disarray has risen. "Kubernetes" is presently some of the time utilized as a shorthand for a whole container condition dependent on Kubernetes. In all actuality, they are not specifically practically identical, have distinctive roots, and tackle for various things. Docker is a stage and apparatus for building, appropriating, and

running Docker containers. It offers its very own local grouping instrument that can be utilized to organize and plan holders on machine group. Kubernetes is a holder arrangement framework for Docker containers that is broader than Docker Swarm and is intended to organize groups of hubs at scale underway in an effective way. It works around the idea of cases, which are booking units (and can contain at least one containers) in the Kubernetes biological community, and they are disseminated among hubs to give high accessibility. One can without much of a stretch run a Docker expand on a Kubernetes group, however Kubernetes itself is certainly not an entire arrangement and is intended to incorporate custom modules. Kubernetes and Docker are both essentially extraordinary advancements, yet they work extremely well together, and both encourage the administration and sending of holders in a dispersed design.

**SECURITY AUDIT & CONCERNS**

Docker Swarm was the most secure way of deployment of the images over the course of time since its release. The way this worked was each node was hardened properly making it nearly impossible to enter the system and destroy it. The Center for Internet Security (CIS) set multiple guidelines on how to harden the Kubernetes or the Docker containers. For instance, one of the recommended practices is to enable built-in Linux security measures, such as SELinux and Seccomp profiles. SELinux is a kernel-level capability that regulates access to files and network resources, while Seccomp profiles restrict the set of system calls an application can make. Together, these capabilities allow a level of fine-grained control over the workloads that run on the node. (TheNewStack, 2018). As a rule, real contemplations of hub security include:

* Anchoring hub correspondences with a TLS customer authentication, to guarantee every single basic Apus passageways are anchored with end-to-end TLS.
* Empowering applicable part level security controls like SELinux or Seccomp. These capacities help to confine the assault surface on the hub, subsequently giving more noteworthy power over security of the whole framework.
* Restricting direct access, e.g., Secure Shell (SSH) access, to Kubernetes hubs: Forcing all entrance to hubs by means of Kubernetes guarantees appropriate access control and logging. This lessens hazard for unapproved access to have assets.
* Pursue industry best practices, for example, CIS Docker Benchmark, to appropriately design and solidify the Linux hubs that run compartments.

The main issue with the container kind of architecture is that the entire cluster depends on this one container. So this in turn means that compromised one container ends up destroying the

entire cluster. According to TechBeacon “The default behavior of many Kubernetes clusters (where a token that provides access to the Kubernetes API mounts into each container) can cause security issues, especially if the token has cluster admin rights. This happens where role-based access control (RBAC) isn’t configured. In this configuration, an attacker who gains access to a single container in the cluster can easily escalate these privileges to gain full control of it.” (TechBeacon, 2018).

The architecture of the Docker Hub is similar to a package repository, with the Docker daemon acting as a package manager on the host. Therefore, it is vulnerable to the same vulnerabilities of package managers. These vulnerabilities include processing, storage and uncompressing of potentially untrusted code, performed by the Docker daemon with root privileges. This code can be either tampered at the source (malicious image) or during the transfer (for instance as a consequence of the –insecure-registry option given to the Docker daemon, that makes possible a Man-in-the-Middle attack between the registry and the host).

**On December 04,2018** the first ever major security vulnerability in the Kubernetes Cluster Orchestration developed by Google Inc. was discovered. The CVE-2018-1002105 was identified carrying a CVSS V3 rating of 9.8 with no special privileges, and a network attack vector. The vulnerability is triggered when specially crafted requests allow users to establish a connection through the Kubernetes API sever to a backend sever. Attackers can use this established channel to execute arbitrary requests on that backend. The below article shows the entire breakdown of the vulnerability.



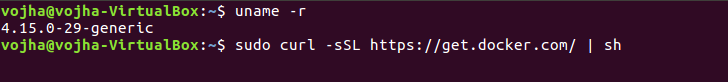
Audit on Ngix demo servers by using Nessus (product of tenable.io).

**Prerequisites**

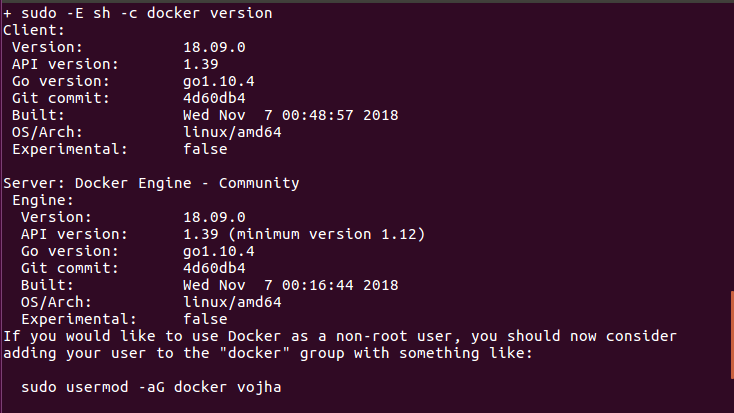
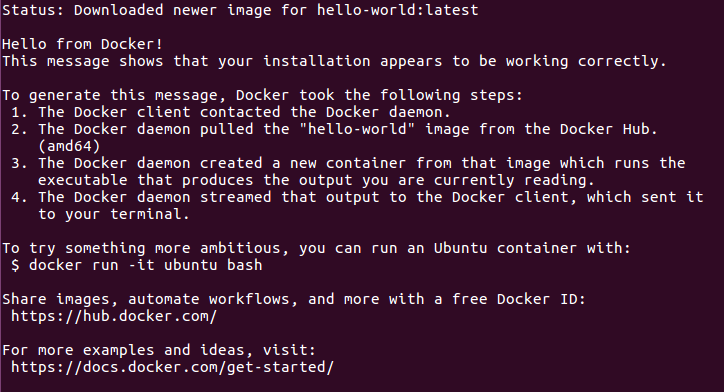
To containerize Nginx, please complete the following:

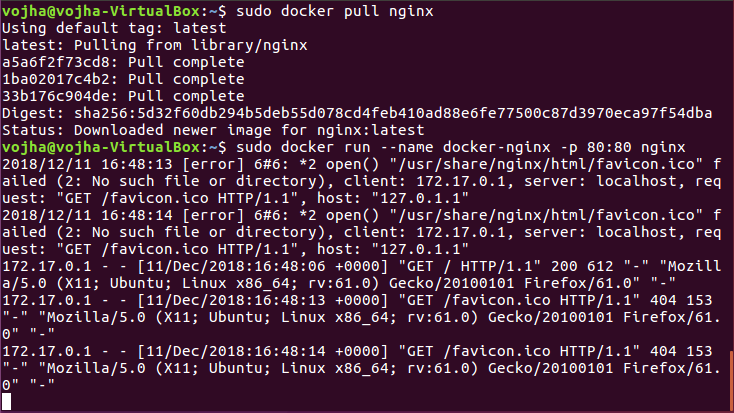
Set up an Ubuntu 14.04 server, preferably with SSH keys for security

Set up a sudo user

Verify your kernel version

Docker hosts a startup script to get Docker up and running on your machine. We can simply run the command: ~$ sudo curl –sSL <https://get.docker.com/> | sh

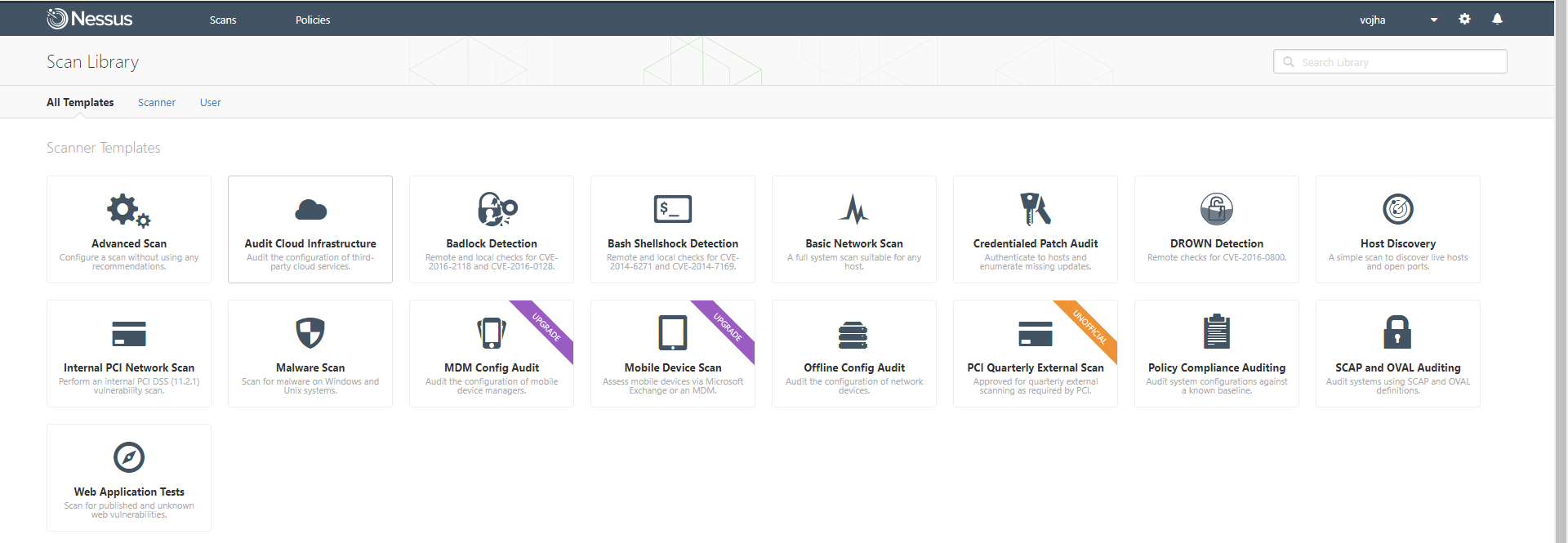




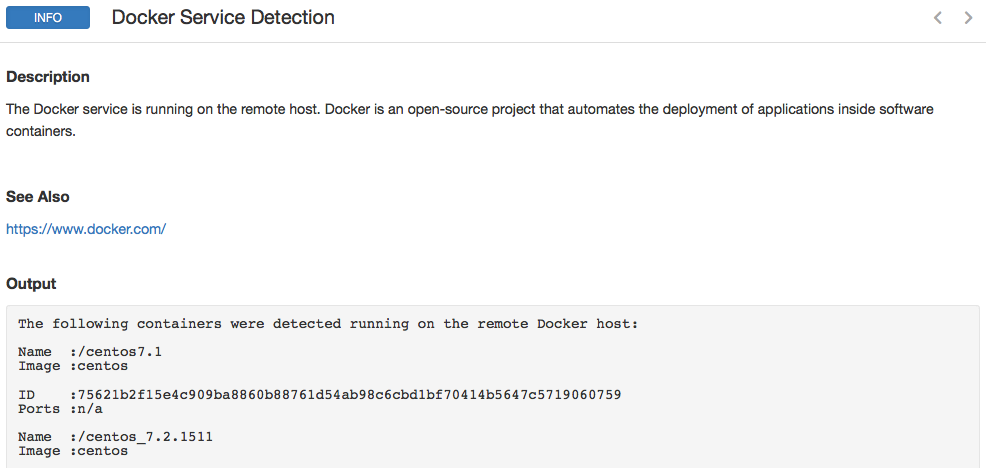
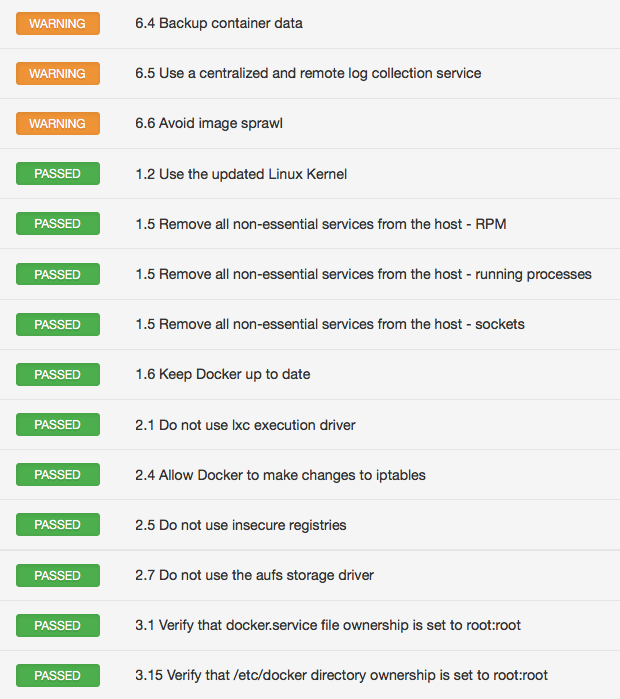
Once the Nginx is up and running on the system you can configure nessus on the same package as the docker

ADD Nessus-7.0.2-ubuntu1110\_amd64.deb /tmp/Nessus-7.0.2-ubuntu1110\_amd64.deb

docker run -d --name nessus-7-ubuntu -p 80:80 fizzymatt/nessus-7-ubuntu

Once configured use Nessus to design a Basic Network scan enabling the Docker Plugins found on tenable.io

Once done, these might be the few outputs generated



\*Results from demo server not actually deployed containers

**Conclusion/Need for further Studies**

Though the new kubernetes, docker container and the containerization process is a better way to deal with the VM Architecture. It sure does has vulnerabilities coming up in the daily for instance on December 04, 2018 a major security flaw was discovered in the Kubernetes. Being closely observed it does appear to be a little complex since required knowledge of Linux scripts and the UNIX environment deployment seems to be a difficult task. Containerization might be the future of the technological world of server shipping’s and images and there is still scope for research and development alongside the tightened security of the docker container & Containerization.

**Research Question**

This study was designed and expected to explore the possibility of changes and level of impact on the software deployment after the discovery and the extensive usage of the Docker Containers. Specifically, the study was expected to answer the question “How useful and easy is it to ship packages using the docker architecture?” This study was conducted under the guidance of people either using the docker containers or are closely related to the security audit features

* What impact has the Kubernetes Orchestration had on the industry running on Hypervisors?
* How can system hardening and the linux server setup help in the deployment of the containers?

**Methodology**

To support and strengthen the research idea this paper used a mixed methods research design. This section generally discusses the selection process involved in the data collection, analysis and conclusion drawing.

**Research Design and Rationale**

This research used the sequential exploratory mixed method of data collection involving the quantitative data survey based on Likert scale followed by a formal focus group study along with a personalized interview with the individuals working in the same field. Using mixed methods research allowed more flexibility, and reliability of the data samples collected than would have a quantitative data based on the Likert scale of the data collection, which would be followed by the personalized reply, by the focus group and personal interviews. Surveys are important because they’re the most reliable method to get real feedback from our subscribers—no matter what platform they use to engage with our brands (print, digital, apps, newsletter, and website) (Estevez, M., 2014). The rationale for mixing both kinds of data within one study is grounded in the fact that neither quantitative nor qualitative methods are sufficient, by themselves, to capture the trends and details of a situation (Creswell, 2006).

**Data Analysis**

The data analysis will involve the analysis of the data obtained from the interview and the survey. This phase will also involve analysis of the earlier survey and data present in various media making comparisons between the survey.

Data collected from interviews and analysis will be used to inform the questions on the survey instrument using descriptive statistics.

**Delimitations**

The project will only focus on the Docker Swarm and Kubernetes Orchestration architecture only and excluding any other docker containers or the shipping software’s that the other docker based containers use which in turn excludes the security features of those containers.

**Appendix A: Definition of Terms**

* **Docker Swarm: “**Docker Swarm or simply Swarm is an open-source container orchestration platform and is the native clustering engine for and by Docker.” (The Newstack).
* **Kubernetes: “**Kubernetes is an open-source platform created by Google for container deployment operations, scaling up and down, and automation across the clusters of hosts.” (The Silicon Valley Business Journal.)
* **Orchestration: “**Orchestration basically comes to automate processes and workflows whereas automation basically automates specific tasks. For example, a deployment process may involve a number of tasks that need to be performed (and even possibly in a specific order - i.e. require statefulness).”(Quora Digest)
* **Vulnerabilities: “**a cyber-security term that refers to a flaw in a system that can leave it open to attack” (Technopedia).

**References**

Creswell, J. W. (2009). Research design: Qualitative, quantitative, and mixed methods approaches. Thousand Oaks, CA: SAGE Publications.

Container Security Considerations in a Kubernetes Deployment. (2018, February 22). Retrieved December 8, 2018, from <https://thenewstack.io/container-security-considerations-kubernetes-deployment/>

DigitalOcean. (2016, October 13). How To Run Nginx in a Docker Container on Ubuntu 14.04. Retrieved December 8, 2018, from <https://www.digitalocean.com/community/tutorials/how-to-run-nginx-in-a-docker-container-on-ubuntu-14-04>

Fattori, A., Lanzi, A., Balzarotti, D., & Kirda, E. (2015). Hypervisor-based malware protection with AccessMiner. Computers & Security, 52, 33-50. doi:10.1016/j.cose.2015.03.007

Layer, B., Omernik, J., Bertucci, Ali, Z., & Constantin, L. (2018, December 04). Critical Vulnerability Uncovered In Kubernetes. Retrieved December 8, 2018, from <https://securityboulevard.com/2018/12/critical-vulnerability-uncovered-in-kubernetes/>

Liggitt, J. (2018, December 04). CVE-2018-1002105: Proxy request handling in kube-apiserver can leave vulnerable TCP connections · Issue #71411 · kubernetes/kubernetes. Retrieved December 8, 2018, from <https://github.com/kubernetes/kubernetes/issues/71411>

McCune, R. (2018, December 05). A hacker's guide to Kubernetes security. Retrieved December 8, 2018, from <https://techbeacon.com/hackers-guide-kubernetes-security>

Mouat, A. (2016, February 05). 5 security concerns when using Docker. Retrieved from <https://www.oreilly.com/ideas/five-security-concerns-when-using-docker>

Martin, A., Raponi, S., Combe, T., & Pietro, R. D. (2018). Docker ecosystem – Vulnerability Analysis. Computer Communications, 122, 30-43. doi:10.1016/j.comcom.2018.03.011

Runtime metrics. (2018, December 05). Retrieved from <https://docs.docker.com/config/containers/runmetrics/>

Revankar, M. (2017, February 08). Auditing Docker with Nessus 6.6. Retrieved December 8, 2018, from <https://www.tenable.com/blog/auditing-docker-with-nessus-66>

Savage, M. (2015, May 07). Top 11 Virtualization Risks Identified. Retrieved December 8, 2018, from <https://www.networkcomputing.com/data-centers/top-11-virtualization-risks-identified/2062567936>

Tozzi, C. (2018, October 22). Docker Security - 6 Ways to Secure Your Docker Containers. Retrieved December 8, 2018, from <https://www.sumologic.com/blog/security/securing-docker-containers/>

Taylor, T. (2018, January 10). Docker Security Issues and Best Practices as We Enter 2018 - DZone Cloud. Retrieved December 8, 2018, from <https://dzone.com/articles/docker-security-issues-and-best-practices-as-we-en>

Vaughan-Nichols, S. J. (2018, March 21). What is Docker and why is it so darn popular? Retrieved December 8, 2018, from <https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/>

**Appendix B: Human Subject Assurances**

Human subjects will be part of this study. People associated with the IT Department dealing with the software development will be a part of the hour long semi-structured interview. The interviews will be closely observed and transcribed. Member checking will ensure that participants could review their information. All participants will have a basic knowledge of the models and thus no prior information or briefing is needed.